

Lecture 6

Using VSAM Datasets

Reviewing Data Sets: When to use what (revisited)

Sequential Data Sets

Disadvantages.

- **Slow** - when the hit rate is low.
- **Complicated to change** (insert, delete, amend)

Advantages.

- **Fast** - when the hit rate is high.
- **Most storage efficient.**
- Simple organization.
- Recovers space from deleted records.

Relative Data Sets

Disadvantages.

- **Wasteful** of storage if the file is only partially populated.
- **Cannot recover space** from deleted records.
- Only a **single, numeric key** allowed.
- Keys must map on to the range of the Relative Record numbers.

Advantages.

- **Fastest** Direct Access organization.
- **Very little storage overhead.**
- Can be read sequentially.

Indexed Data Sets

Disadvantages.

- **Slowest** Direct Access organization.
- **Especially slow** when adding or deleting records.
- **Not very storage efficient.** Must store the Index records, the alternate Index records, the data records and the alternate data records.

Advantages.

- Can use **multiple, alphanumeric keys.**
- Can have **duplicate alternate keys.**
- Can be **read sequentially** on any of its keys.
- Can **partially recover space** from deleted records.

Comparing Access Methods for Different File Organizations

		Access Method		
		Sequential	Random	Dynamic
Data Organization	Sequential	Accessed Serially	NA	NA
	Relative	Accessed in record number order	Specify a record number to get the data record	Can access sequentially or randomly
	Indexed	Accessed in primary key order	Specify a key to get the data record	

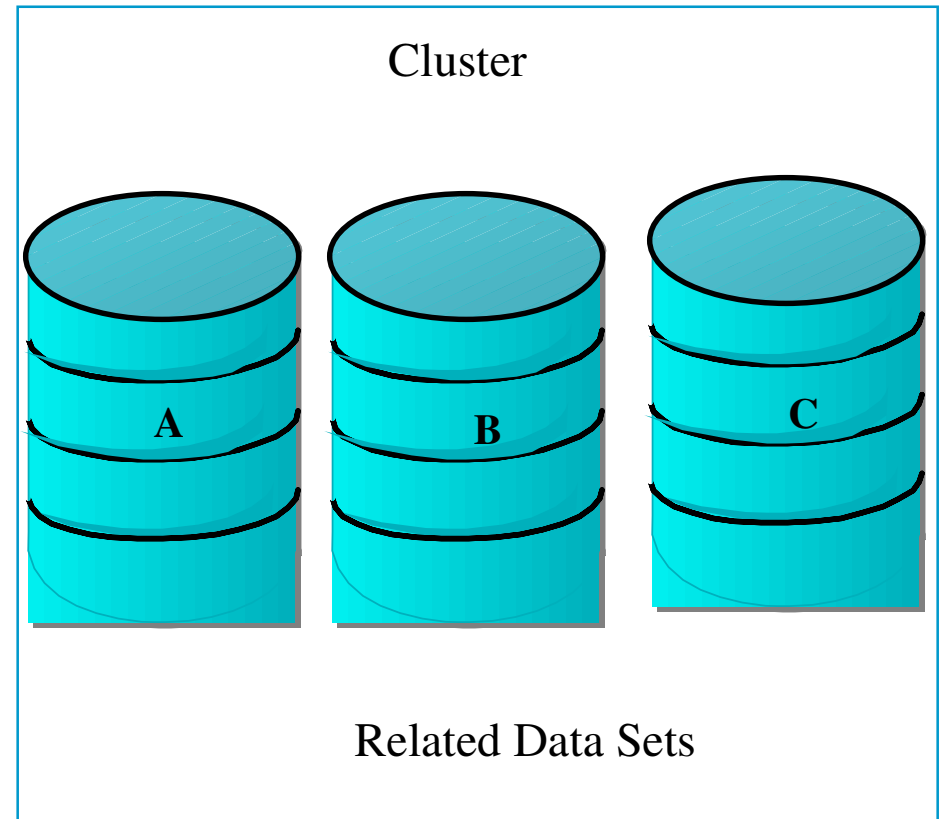
VSAM

Virtual Storage Access Method

Cluster

A cluster is the collection of physical data sets that make up one logical data set. The concept of a cluster is more suited for a KSDS.

- A KSDS (Key Sequences Data Set) cluster has two data sets. One data set holds the actual data records. The other data set contains an index component.
- The index component permits the direct retrieval of data.



Advantages and Limitations of VSAM

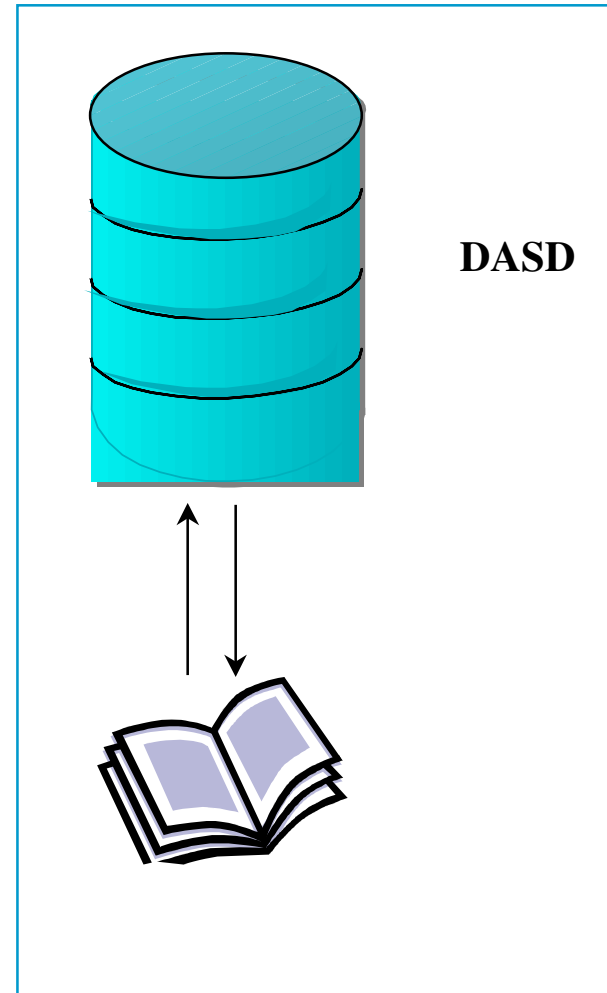
- The advantages of VSAM are:
 - VSAM supports more data set types
 - Simplifies record processing
 - Supports a variety of I/O techniques
 - Provides greater efficiency for the application programs and for the overall environment
- The major limitation of VSAM is:
 - Its data sets must reside on DASD. They cannot be created on tape.

Allocating a data set

- To use a data set, you first *allocate* it.
- Ways to allocate a data set:
 - ISPF data set panel, option 3.2
 - TSO ALLOCATE command
 - Job Control Language (JCL)
 - Access Method Services
(IDCAMS - Integrated Catalog Access Method Services)
- Then, access the data using the access method that you have chosen.

Defining a VSAM Data Set

- The process of creating catalog entries for a VSAM data set and allocating space for them is called defining the data set.
- A data set must be defined before it is loaded with data or accessed by a processing program.
- VSAM data sets can be defined using either IDCAMS or JCL.



An Example of Using An Indexed Dataset (KSDS)

Creating a VSAM Dataset Using JCL

```
//CREATEKS JOB
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DEFINE CLUSTER (NAME(KC02170.DATATEST.T08) -
               RECORDS(15 15) -
               RECORDSIZE(71 71) -
               KEYS(11 0) -
               INDEXED) -
      DATA (NAME(KC02170.DATATEST.T08.DAT)) -
      INDEX (NAME(KC02170.DATATEST.T08.IDX))
/*
//*
```

Key is 11 bytes, starting at location 0

The average record size is 71 bytes (as well as the max)

15 Records are initially created

Deleting a VSAM Dataset Using JCL

```
000112 //DELVSAM JOB
000120 //STEP1  EXEC PGM=IDCAMS
000130 //SYSPRINT DD SYSOUT=*
000140 //SYSIN   DD *
000400  DELETE  KC02170.DATATEST.V80
000900 /*
001000 //
```

Adding an Account - initial definitions

000700* file for Reading new account to process

000800 Select AcctsFD

000900 Assign to "Accts"

001000 Access mode is RANDOM

001100 Organization is INDEXED

001110 Record Key is accountNum of AccountData

001200 File Status is AcctsStatus.

001210*

001300 Data Division.

001400 File Section.

001500 FD AcctsFD.

001700 01 AccountData.

001800 05 creditLimit PIC 9(5).

001900 05 accountNum PIC X(6).

001910 05 Name PIC X(10).

001920 05 FILLER PIC X(59).

Adding an Account - Procedure

```
003100 Procedure division.  
003300   perform openAccountData.  
003301*  
003302   DISPLAY "Enter new account: "  
003303   DISPLAY "   Limit 9(5), AccountNum x(6), name X(10)"  
003304   ACCEPT NewAccount.  
003310*  
003320   move name of newAccount to name of AccountData.  
003330   move creditLimit of newAccount to creditLimit of AccountData.  
003340   move accountNum of newAccount to accountNum of AccountData.  
003400   write AccountData.  
003700   if AcctsStatus is not equal to "00"  
003800       Display "write Error for Acct " AcctsStatus  
003900   end-if.  
004000  
004110   perform closeAccountData.  
004200   stop run.
```


Adding an Account - Open/Close

```
004300 OpenAccountData.  
004400     Display "trying to open AcctMast"  
004500     open I-O AcctsFD.  
004600     if AcctsStatus is not equal to "00"  
004700         Display "open Error for Account Status " AcctsStatus  
004800     end-if.  
004900     exit.  
005000*  
005100 CloseAccountData.  
005200     close AcctsFD.  
005300     if AcctsStatus is not equal to "00"  
005400         Display "close Error for AcctMast " AcctsStatus  
005500     end-if.  
005600     Display "done closing AccMast"  
005700     exit.
```

Print all Accounts

Print All Accounts - Initial Definitions

000700* file for Reading new account to process

000800 Select optional AcctsFD

000900 Assign to "Accts"

001000 Access mode is SEQUENTIAL

001100 Organization is INDEXED

001110 Record Key is AccountNum of AccountData

001200 File Status is AcctsStatus.

001210*

001300 Data Division.

001400 File Section.

001500 FD AcctsFD.

001700 01 AccountData.

001800 05 creditLimit PIC 9(5).

001900 05 accountNum PIC X(6).

001910 05 Name PIC X(10).

001920 05 FILLER PIC X(59).

Print All Accounts - Loop through

```
003100 Procedure division.  
003200  
003300     perform openAccountData.  
003301*  
003302     move 1 to moreAccounts.  
003304     perform until moreAccounts = 0  
003306         read acctsFD  
003307             at end move 0 to moreAccounts  
003308             not at end perform outputAccount  
003309     end-read  
003320     if AcctsStatus is not equal to "00"  
003330         Display "read Error for AcctMast " AcctsStatus  
003340         move 0 to moreAccounts  
003350     end-if  
003370 end-perform.  
004110 perform closeAccountData.  
004200 stop run.
```

Print All Accounts - output a Single Account

```
005800*-----  
005900* output a record in the account master  
006100*  
006200 OutputAccount.  
006300     Display "Account Number: " AccountNum.  
006400     Display "   Credit Limit: " creditLimit.  
006500     Display "   Name: " Name.  
006800     exit.
```

Printing a Specific Account

Printing an Account

```
003100 Procedure division.  
003200  
003300    perform openAccountData.  
003301*  
003302    DISPLAY "ENTER ACCOUNT NUMBER X(6)"  
003303    ACCEPT InputAccountNumber.  
003304    move InputAccountNumber to AccountNum.  
003305*  
003306    read acctsFD  
003320    if AcctsStatus is not equal to "00"  
003330        Display "read Error for AcctMast " AcctsStatus  
003340    else  
003341        Perform OutputAccount  
003350    end-if  
004110    perform closeAccountData.  
004200    stop run.
```

For your Reference - Indexed File IO Statements

A Select statement for sequential Access

```
000220 Input-output section.
000221 File-control.
000222*
000223*-----
000224* file for storing the next available number
000225     Select NxtAccNFD
000226     Assign to "NxtAccN"
000227     Access mode is sequential
000228     Organization is sequential
000229     File Status is NxtAccNStatus.
000230*-----
000235 Data Division.
000240 File Section.
000241 FD NxtAccNFD
000243 01 AccountNumberRecord.
000244     05 accountNumber PIC 9(5).
```

A Select statement for Random Access

```
000231 *-----
000232 * storing the Account Master Information
000233     Select AcctMastFD
000234     Assign to "AcctMast"
000235     Access mode is Random
000236     Organization is indexed
000237     Record Key is SSN of AccountMasterData
000238     File Status is AcctMastStatus.
000239 *-----
000247 Data Division.
000248 File Section.
000254 FD AcctMastFD
000270 01 AccountMasterData.
000279 05 SSN.
000285 10...
```

The syntax of the Select statement for indexed files

```
SELECT file-name ASSIGN to system-name  
          ORGANIZATION IS INDEXED  
          [ ACCESS MODE IS { SEQUENTIAL  
                               RANDOM  
                               DYNAMIC } ]  
          RECORD KEY IS data-name-1  
          [ FILE STATUS IS data-name-2 ]
```

The syntax for a system name for a VSAM file on an IBM mainframe

ddname

The Open statement for indexed files

$$\underline{\text{OPEN}} \left\{ \begin{array}{l} \underline{\text{INPUT}} \text{ file-name-1} \dots \\ \underline{\text{OUTPUT}} \text{ file-name-2} \dots \\ \underline{\text{I-O}} \text{ file-name-3} \dots \\ \underline{\text{EXTEND}} \text{ file-name-4} \dots \end{array} \right\} \dots$$

Note

- You can open a file in extend mode only if sequential access is specified.

The Read statement for indexed files

The Read statement for sequential access

```
READ file-name [NEXT] RECORD [INTO data-name]  
      [AT END imperative-statement-1]  
      [NOT AT END imperative-statement-2]  
      [END-READ]
```

- If a file is opened for dynamic access, you must include the word NEXT to access the records sequentially. Otherwise, random access is assumed.
- If a file is accessed sequentially, you can include the At End and Not At End clauses.

The Read statement for indexed files

The Read statement for random access

```
READ file-name RECORD [INTO data-name]  
      [INVALID KEY imperative-statement-1]  
      [NOT INVALID KEY imperative-statement-2]  
      [END-READ]
```

- If a file is opened for dynamic access, you must include the word NEXT to access the records sequentially. Otherwise, random access is assumed.
- If the file is accessed randomly, you can use the Invalid Key and Not Invalid Key clauses.
- The Invalid Key clause is executed whenever an invalid key condition occurs.
- If an invalid key condition doesn't occur, the Not Invalid Key clause is executed.

The Rewrite statement for indexed files

The Rewrite statement

```
REWRITE record-name [FROM data-name]  
    [INVALID KEY imperative-statement-1]  
    [NOT INVALID KEY imperative-statement-2]  
    [END-REWRITE]
```

Description

- If a file is opened for sequential access, the Rewrite statement can be used only after executing a Read statement for the record to be rewritten.
- If the file is opened for random or dynamic access, you can use Rewrite without a prior Read.
- The Invalid Key clause is executed whenever an invalid key condition occurs.
- If an invalid key condition doesn't occur, the Not Invalid Key clause is executed.

The Write and Close statements for indexed files

The Write statement

```
WRITE record-name [FROM data-name]  
          [INVALID KEY imperative-statement-1]  
          [NOT INVALID KEY imperative-statement-2]  
          [END-WRITE]
```

The Close statement

```
CLOSE file-name-1 ...
```

Description

- The Invalid Key clause of the Write statement is executed whenever an invalid key condition occurs.
- If an invalid key condition doesn't occur, the Not Invalid Key clause is executed.

The syntax of the Start statement

<u>START</u> file-name	KEY IS	<table border="0"><tr><td><u>EQUAL TO</u></td></tr><tr><td>=</td></tr><tr><td><u>GREATER THAN</u></td></tr><tr><td>></td></tr><tr><td><u>NOT LESS THAN</u></td></tr><tr><td><u>NOT</u> <</td></tr><tr><td><u>GREATER THAN OR EQUAL TO</u></td></tr><tr><td>>=</td></tr></table>	<u>EQUAL TO</u>	=	<u>GREATER THAN</u>	>	<u>NOT LESS THAN</u>	<u>NOT</u> <	<u>GREATER THAN OR EQUAL TO</u>	>=	data-name
<u>EQUAL TO</u>											
=											
<u>GREATER THAN</u>											
>											
<u>NOT LESS THAN</u>											
<u>NOT</u> <											
<u>GREATER THAN OR EQUAL TO</u>											
>=											

[INVALID KEY imperative-statement-1]
[NOT INVALID KEY imperative-statement-2]
[END-START]

A Start statement that positions the file at the record with the specified key value

```
MOVE "1000" TO IM-ITEM-NO.  
START INVMAST  
    KEY IS >= IM-ITEM-NO.
```

How to code the Start statement

- You can use the Start statement to position an indexed file at a specific location based on a key value.
- Once the file is positioned at the correct record, you can use the Read statement to access records sequentially starting with that record.
- You can use the Start statement with a file that's opened for sequential or dynamic access.

The syntax of the Delete statement

```
DELETE file-name RECORD  
    [INVALID KEY imperative-statement-1]  
    [NOT INVALID KEY imperative-statement-2]  
    [END-DELETE]
```

A Delete statement that deletes a record using random access

```
DELETE INVMAS  
    INVALID KEY  
        DISPLAY "INVALID KEY ON DELETE FOR ITEM NUMBER "  
            IM-ITEM-NO.
```

How to code the Delete statement

- The Delete statement can be used only for a file that's opened in I-O mode.
- If a file has sequential access, the Delete statement can be used only after executing a Read statement for the record to be deleted. In that case, the Invalid Key and Not Invalid Key clauses should not be specified.
- If a file has random or dynamic access, the Delete statement can be used without reading the record first. In that case, the Invalid Key clause should be specified.